SECTION IV

ROM AND DOS ROUTINES

SECTION IV

ROM AND DOS ROUTINES

In an assembly language program, the simplest way to use the I/O devices is with ROM and DOS routines. This section shows how.

Complete lists of the ROM routines and DOS routines are in the reference section.

Chapter 13/ Using the Keyboard and Video Display (ROM Routines)

The Color Computer uses its own machine-code routines to access the screen, keyboard, and tape. These routines are built into the computer's ROM. You can use the same routines in your own program.

Appendix F lists each ROM routine and the ROM address that points to it. This chapter uses two of these routines, POLCAT and CHROUT, as samples in showing the steps for using ROM routines.

Steps for Calling ROM Routines

We recommend these steps for calling a ROM routine:

1. Equate the routine's address to its name. This lets you refer to the routine by its name rather than its address, making your program easier to read and revise.
2. Set up any entry conditions required by the routine. This lets you pass data to the routine.
3. Preserve the contents of the registers. Since many routines change the contents of the registers, you might want to store the registers' contents temporarily before jumping to the routine.
4. Call the ROM routine, using the indirect addressing mode.
5. Use any exit conditions that the routine passes back to your program.
6. Restore the contents of the registers (if you temporarily preserved them in Step 3).

Sample 1 Keyboard Input with POLCAT

POLCAT "polls" the keyboard to see if you press a key. If you do not, POLCAT sets Bit Z.

If you do press a key, POLCAT:

(1) Clears Bit Z of Register CC and
(2) Loads Register A with the key's ASCII code.

This short program uses POLCAT to poll the keyboard. When you press a key, the program ends:

```
            ORG     $1200
BEGIN       JMP     START
            FDB     DONE-BEGIN
POLCAT      EQU     $A000
START       PSHS    DP,CC,X,Y,U
WAIT        JSR     [POLCAT]
            BEQ     WAIT
            PULS    DP,CC,X,Y,U
            CLR     $71
            JMP     [$FFFE]
DONE        *
            END
```

This is how we applied the above steps in writing this program:

1. Equate POLCAT to its Address

This equates POLCAT to $A000, the address that points to POLCAT's address:

```
POLCAT      EQU     $A000
```

2. Set Up Entry Conditions

POLCAT has no entry conditions.

3. Preserve the Registers' Contents

POLCAT's "Exit Conditions" state that POLCAT modifies all registers except B and X. Assume that you want to preserve the contents of Registers DP, CC, X, Y, and U. To do this, you can "push" these values into the "hardware stack":

```
PSHS    DP,CC,X,Y,U
```

(The hardware stack is an area of memory, pointed to by Register S, that the processor uses for subroutines. PSHS "preserves" the contents of certain registers by storing them in the hardware stack.)

4. Jump to POLCAT

This jumps to POLCAT using its indirect address:

```
WAIT    JSR    [POLCAT]
```

5. Use Exit Conditions

For now, assume you want to look only at the status of Bit Z to see if a key has been pressed:

```
BEQ    WAIT
```

The above instruction branches back to WAIT (the JSR [POLCAT] instruction) unless you press a key. (Pressing a key causes POLCAT to clear Bit Z.)

6. Restore the Register's Contents

This "pulls" (inserts) the contents of the hardware stack back into the registers:

```
PULS    DP,CC,X,Y,U
```

Now, the above registers are restored to the data they contained before executing the POLCAT routine.

Sample 2
Character Output with CHROUT

The CHROUT routine prints a character on either the screen or printer. On entry, it checks two places:

- Register A — to determine which character to print
- Address $6F — to determine whether to print it on the screen or the printer

This program uses CHROUT to print "This is a Message" on the screen. It then uses POLCAT to wait for you to press a key before returning to BASIC.

```
          ORG    $1200
****** Equates for Routines ******
POLCAT    EQU    $A000
CHROUT    EQU    $A002
DEVNUM    EQU    $6F
************ Variable ************
SCREEN    EQU    00
*** DOS Programming Convention ***
BEGIN     JMP    START
          FDB    DONE-BEGIN
******** Print the Message ********
START     LDB    #SCREEN
          STB    DEVNUM
          LDX    #MSG
PRINT     LDA    ,X+
          JSR    [CHROUT]
          CMPA   #$0D
          BNE    PRINT
********* Wait for a Key *********
INPUT     PSHS   DP,CC,X,Y,U
WAIT      JSR    [POLCAT]
          BEQ    WAIT
          PULS   DP,CC,X,Y,U
          CLR    $71
          JMP    [$FFFE]
************* Message *************
MSG       FCC    'THIS IS A MESSAGE'
          FCB    $0D
******** Memory for Stack ********
DONE      *
          END
```

Most of the steps we used in writing this program are obvious. What may not be obvious is the way we set up CHROUT's entry conditions, Address $6F and Register A.

These lines set Address $6F to 00 (the screen):

```
DEVNUM    EQU    $6F
SCREEN    EQU    00
START     LDB    #SCREEN
          STB    DEVNUM
```

Setting Register A involves two steps. First, point Register X to the message:

```
MSG     FCC   'THIS IS A MESSAGE'
        FCB   $ØD
        LDX   #MSG
```

and then load Register A with each character in the message:

```
PRINT   LDA   ,X+
        JSR   [CHROUT]
        CMPA  #$ØD
        BNE   PRINT
```

Sample 3
POLCAT and CHROUT

This combines POLCAT with CHROUT. It prints on the screen whatever key you press. When you press ↓ (hexadecimal 0A), the program returns to BASIC:

```
          ORG   $12ØØ
****** Equates for Routines ******
POLCAT    EQU   $AØØØ
CHROUT    EQU   $AØØ2
DEVNUM    EQU   $6F
************ Variable ************
SCREEN    EQU   ØØ
*** DOS Programming Convention ***
BEGIN     JMP   MAIN
          FDB   DONE-BEGIN
********** Main Program **********
MAIN      JSR   INPUT
          CMPA  #$ØA
          BEQ   FINISH
          JSR   PRINT
          BRA   MAIN
FINISH    CLR   $71
          JMP   [$FFFE]
* Input a Character from Keyboard *
INPUT     PSHS  DP,CC,X,Y,U
WAIT      JSR   [POLCAT]
          BEQ   WAIT
          PULS  DP,CC,X,Y,U
          RTS
** Print a Character on Display **
PRINT     LDB   #SCREEN
          STB   DEVNUM
          JSR   [CHROUT]
          RTS
******** Memory for Stack ********
DONE      *
          END
```

Chapter 14/ Opening and Closing a Disk File DOS Routines — Part I

Because of the organization and timing of a disk, reading it and writing to it are complex. This is why you'll want to make use of DOS routines in your disk programs.

This chapter shows how to use DOS routines to open and close a disk file. The next chapter shows how to use them to read a disk and write to it. *Reference H* contains a complete list of all the DOS routines supported by Radio Shack.

Overview

All DOS routines, like ROM routines, have their own entry and exit conditions. However, most DOS routines have more involved entry conditions than do ROM routines. They require you to set up three areas in memory: two "buffers" and a "data control block."

Buffers

Buffers are areas in memory that DOS uses for storing data to be input or output to disk. DOS requires that you reserve two buffers:

- A logical buffer — This can be any length. Your program uses this to store data for DOS to input or output to disk.
- A physical buffer — This must be 256 bytes. DOS uses this to hold data temporarily so that it can input and output the data to a disk sector in 256-byte blocks.

For example, suppose you want to output 100 10-byte records to disk. You can send each record, one at a time, to the area you reserved as the logical buffer.

DOS then transfers the records from the logical buffer to the area you reserved as the physical buffer. As soon as there are 256 bytes in the physical buffer, DOS sends them out to a disk sector.

You need not be concerned that DOS' "physical" records are a different size from your program's "logical" records. DOS handles the "spanning" of logical records into physical records internally. Except for reserving memory for a physical buffer, you do not need to be concerned with physical records.

Data Control Block

A data control block is a 49-byte "block" of memory that DOS uses to control a disk file. You need to reserve this block of memory for each disk file you are using. If you have three disk files open at the same time, you need to reserve three 49-byte data control blocks.

Reference G shows how DOS uses each of the 49 bytes, numbered 0-48, in the data control block. As you can see, DOS divides the data control block into 21 data-control segments.

Before opening a file, you must load the proper data into four of the segments of the data control block (DCB):

DCB Segment	DCB Address	You must load with . . .
Filename (DCBFNM)	Bytes 0-7	The eight-character name of your file.
Extension (DCBEXT)	Bytes 8-10	The three character extension of your filename.
Drive Number (DCBDRV)	Byte 33	The drive containing the disk file.

Physical Buffer Address (DCBBUF)	Byte 36-37	The first address of the physical buffer you have reserved.

For example, if you want to open a file in Drive 1, you need to load "1" into the DCBDRV location, which is the 33rd byte of the data control block.

You need not be concerned with most of the remaining segments of the data control block, unless you want to use them as data in your program. They are handled internally by DOS. The exceptions to this are:

- Logical Buffer Address, Record Size, Variable Record Terminator, and Logical Record Number — You need to use these when you read and write to the file. They are discussed in the next chapter.
- *File Type* and *ASCII Flag* — If you want your file to be compatible with BASIC and other Radio Shack programs, you need to set these when you create the file. See the "Technical Information" chapter of your *Disk System Owners Manual and Programming Guide*.

Steps for Using DOS Routines

The steps for using DOS routines are:

1. Equate the routine's address (for ease in reading the program).
2. Reserve memory for a physical buffer, logical buffer, and the DCB.
3. Clear the DCB and the physical buffer. You need to make sure they do not have extraneous data.
4. Set up all other entry conditions. Besides setting up registers, you need to load certain segments of the DCB with data. Which segments you load depends on the DOS routine you are using.
5. Preserve the contents of the registers. DOS routines change the contents of many of the registers. To be safe, you should preserve all of them that you want to use later in your program. Be sure to preserve Registers U and DP. If DOS changes their contents, your program acts unpredictably.
6. Call the routine.
7. Restore the contents of the registers.
8. Use all exit conditions. Most DOS routines return an error code in Register A if the routine did not work properly. If there were no errors, Register A contains a zero.

Sample Session Opening and Closing a Disk File

The DOS routines for opening and closing a file are OPEN and CLOSE. Both routines check Register U for the address of DCB. They expect to find the four segments described above in this block.

OPEN also expects you to set a file mode in Register A. It creates or opens an existing file depending on the mode you set.

Both routines return a status code in Register A. *Reference I* tells the meaning of the status codes.

Figure 8 at the end of this chapter is a sample program which creates, opens, and closes a disk file named WORKFILE/TXT. After running this program, you can look at your directory to see that the program has created this file. This shows how we applied the above steps in this program.

1. Equate OPEN and CLOSE

This equates OPEN and CLOSE to $600 and $602, their indirect addresses:

```
OPEN        EQU         $600
CLOSE       EQU         $602
```

2. Reserve Memory for Buffers and DCB

The OPEN and CLOSE routines use only the physical buffer, not the logical buffer. This stores 256 bytes for the physical buffer and uses PBUF to label those bytes:

```
PBUF        RMB         256
```

This reserves memory for a 49-byte DCB and stores the filename, WORKFILE, and the extension, TXT, in the first 11 bytes:

```
DCB         EQU         *
            FCC         'WORKFILE'
            FCC         'TXT'
            RMB         38
```

3. Clear DCB

This clears all but the first 11 bytes of DCB:

```
RCLEAR    LDX     #DCB+11
CLEAR1    CLR     ,X+
          CMPX    #DCB+48
          BNE     CLEAR1
          LDX     #PBUF
```

and this clears the physical buffer:

```
CLEAR2    CLR     ,X+
          CMPX    #PBUF+255
          BNE     CLEAR2
          RTS
```

4. Set Up Entry Conditions

On entry, OPEN and CLOSE require you to: (1) Set Register U to a DCB containing a filename, extension, drive number, and physical buffer address, and (2) Set Register A to a file mode.

Setting Register U

This sets Register U to the address of the first byte of the DCB:

```
          LDU     #DCB
```

The following lines set the drive number segment to 0. They do this by storing DRVNUM (0) into DCBDRV (33) + the contents of Register U (DCB). This inserts 0 into the 33rd byte of DCB:

```
DCBDRV    EQU     33
DRVNUM    FCB     00
          LDA     DRVNUM
          STA     DCBDRV,U
```

The following lines set the physical buffer address to PBUF. They do this by storing the address of PBUF into the memory address pointed to by Register U plus DCBBUF. This stores PBUF in the 36th byte of DCB:

```
DCBBUF    EQU     36
          LDX     #PBUF
          STX     DCBBUF,U
```

(The filename and extension were set in Step 2.)

Setting Register A

This table shows how you should set each bit in Register A to select one or more file modes:

MODE	BIT	DECIMAL NUMBER (IF SET)
Read	Bit 0	1
Write	Bit 1	2
Create	Bit 2	4
Extend	Bit 3	8
Work File (delete the file, when closed)	Bit 4	16
FAT (rewrite to the FAT* only when closed)	Bit 5	32
Shared Buffer	Bit 6	64

* The disk directory's FAT (file allocation table) is described in the "Technical Information" chapter of the *Disk System Manual*.

The sample program loads Register A with decimal 1+2+4+8+32:

```
          LDA     #1+2+4+8+32
```

This tells DOS to set the file mode to read (decimal 1), write (decimal 2), create (decimal 4), extend (decimal 8), and rewrite the FAT only when the file is closed (decimal 32).

5. Preserve Registers

This preserves the contents of Registers U and DP:

```
ROPEN     PSHS    U,DP
```

6. Jump to the DOS Routine

These lines jump to OPEN and CLOSE:

```
          JSR     [OPEN]
          JSR     [CLOSE]
```

7. Restore Registers

This restores the contents of Registers U and DP:

```
          PULS    U,DP
```

8. Use Exit Conditions

The sample program branches to an error handling subroutine after each DOS routine. The subroutine tests Register A to see if it contains a non-zero value. If so, it

prints the status code on the screen and waits for you to press a key:

```
                JSR         ERROR
                TSTA
                BEQ         RETURN
                STA         $450
    WAIT        JSR         [POLCAT]
                BEQ         WAIT
                RETURN      RTS
```

Figure 8. Sample Program to Open and Close a File

```
                ORG         $1200
**Equates for DOS and ROM routines **
    OPEN        EQU         $600
    CLOSE       EQU         $602
    POLCAT      EQU         $A000
****** Equates for DCB offsets *******
    DCBDRV      EQU         33
    DCBBUF      EQU         36
*****DOS Programming Convention *****
    BEGIN       JMP         MAIN
                FDB         DONE-BEGIN
************Main Program ************
    MAIN        JSR         RCLEAR
                JSR         ROPEN
                JSR         RCLOSE
                CLR         $71
                JMP         [$FFFE]
******Routine to Clear the DCB ******
******** and Physical Buffer*********
    RCLEAR      LDX         #DCB+11
    CLEAR1      CLR         ,X+
                CMPX        #DCB+48
                BNE         CLEAR1
                LDX         #PBUF
    CLEAR2      CLR         ,X+
                CMPX        #PBUF+255
                BNE         CLEAR2
                RTS
*******Routine to Open a File *******
    ROPEN       PSHS        U,DP
                LDU         #DCB
                LDA         DRVNUM
                STA         DCBDRV,U
                LDX         #PBUF
                STX         DCBBUF,U
                LDA         #1+2+4+8+32
                JSR         [OPEN]
                PULS        U,DP
                JSR         ERROR
                RTS
***** Routine to Close the File ******
    RCLOSE      PSHS        U,DP
                LDU         #DCB
                JSR         [CLOSE]
                PULS        U,DP
                JSR         ERROR
                RTS
*******Error Handling Routine *******
    ERROR       TSTA
                BEQ         RETURN
                STA         $450
    WAIT        JSR         [POLCAT]
                BEQ         WAIT
    RETURN      RTS
*** Memory for Buffers and Stacks ****
    PBUF        RMB         256
********Memory for Variables ********
    DRVNUM      FCB         00
***********Memory for DCB ***********
    DCB         EQU         *
                FCC         'WORKFILE'
                FCC         'TXT'
                RMB         38
************************************
    DONE        EQU         *
                END
```

Chapter 15/ Reading and Writing a Disk File DOS Routines — Part 2

DOS has a WRITE routine for writing to a file and a READ routine for reading it back into memory. The way you use these routines depends on which method you are using to access the file:

- Sequential Access
- Direct Access

This chapter describes how to use these two methods in their simplest forms. You can use any variation of them that you want.

Sequential vs. Direct Access

Sequential Access
(For Files with Variable-Length Records)

Sequential access lets you read and write to files with variable-length records. Using this method, you insert a terminator character at the end of each record. This character tells DOS where each record ends.

Before writing data to the file, you must load DCB with the following:

DCB Segment	DCB Address	You must load with. . .
Logical Buffer Address (DCBLRB)	Bytes 39-40	The first address of the logical buffer you have reserved
Terminator Character (DCBTRM)	Byte 19	The character you select to end each record

When reading data from just one file, you need only specify the logical buffer address, not the terminator character. DOS reads the terminator character from the disk's directory into DCBTRM.

Figure 9 at the end of this chapter is a program that writes to a file using $0D (the ENTER character) as a terminator character. *Figure 10* reads the same file back into memory.

Direct Access
(For Files with Fixed-Length Records)

Direct access works only with files containing fixed-length records. With this method, DOS uses the record size and record number to access the record.

Before reading data from the file or writing data to it, you must set this DCB segment:

DCB Segment	DCB Address	You must load with . .
Logical Buffer Address (DCBLRB)	Bytes 39-40	The address of the first byte of the logical buffer you have reserved

Unless you are using the record size already in the file's directory, you must also set:

Logical Record Size (DCBRSZ)	Bytes 17-18	The size of each record

If you want to write a record which is not sequentially the next one, you must also set:

Logical Record Number (DCBLRN)	Bytes 46-47	The number of the record you want to access

Setting the Read/Write Option

DOS requires that you set Register A with a "read/write option" before entering the READ or WRITE routines. The read/write option lets you specify:

- Whether you want direct or sequential access
- Whether you want DOS to point to the next record after reading or writing the record

To set the read/write option, load the first two bits of Register A with one of these four values:

Read/Write Option	Bits	Decimal Number
Direct Access Point to next record	00	0
Sequential Access Point to next record	01	1
Direct Access Do not point to next record	10	2
Sequential Access Do not point to next record	11	3

For example:

```
LDA         #2
JSR         [READ]
```

tells DOS to write the record sequentially (up to the terminator character). When finished, DOS points to the next sequential record.

Figure 9. Sample Program to Write to a File

```
            ORG         $1200
 **Equates for DOS and ROM routines **
OPEN        EQU         $600
CLOSE       EQU         $602
WRITE       EQU         $606
POLCAT      EQU         $A000
 ****** Equates for DCB offsets *******
DCBTRM      EQU         19
DCBDRV      EQU         33
DCBBUF      EQU         36
DCBLRB      EQU         39
 *****DOS Programming Convention *****
BEGIN       JMP         MAIN
            FDB         DONE-BEGIN
 ************Main Program ************
MAIN        JSR         CLEAR
            JSR         INTDCB
            JSR         SOPEN
            JSR         SPRINT
            JSR         SWRITE
            JSR         SCLOSE
            CLR         $71
            JMP         [$FFFE]
 ******Routine to Clear the DCB ******
 and the Physical and Logical Buffers
CLEAR       LDX         #PBUF
CLEAR1      CLR         ,X+
            CMPX        #PBUF+255
            BNE         CLEAR1
            LDX         #LBUF
CLEAR2      CLR         ,X+
            CMPX        #LBUF+24
            BNE         CLEAR2
            LDX         #DCB+11
CLEAR3      CLR         ,X+
            CMPX        #DCB+48
            BNE         CLEAR3
            RTS
 ********* Routine to Insert **********
 ********* Values in the DCB **********
INTDCB      LDU         #DCB
            LDA         DRVNUM
            STA         DCBDRV,U
            LDA         #$0D
            STA         DCBTRM,U
            LDX         #PBUF
            STX         DCBBUF,U
            LDX         #LBUF
            STX         DCBLRB,U
            RTS
 *******Routine to Open a File *******
SOPEN       LDU         #DCB
            PSHS        U,DP
            LDA         #1+2+4+8+32
            JSR         [OPEN]
            PULS        U,DP
            JSR         ERROR
            RTS
 ********Routine to Print Msg ********
```

```
SPRINT    LDY       #$500
          LDX       #MSG
CHAR      LDA       ,X+
          STA       ,Y+
          CMPA      #$3A
          BNE       CHAR
          LDX       #LBUF
          LDY       #$525
 ******* Routine to Input Data ********
 *********** from Keyboard ************
SINPUT    PSHS      U,DP,Y
WAIT1     JSR       [POLCAT]
          BEQ       WAIT1
          PULS      U,DP,Y
          STA       ,Y+
          STA       ,X+
          CMPA      #$0D
          BEQ       ENDINP
          CMPX      #LBUF+24
          BNE       SINPUT
ENDINP    RTS
 ******* Routine to Write Data ********
 *************** to File ***************
SWRITE    PSHS      U,DP
          LDU       #DCB
          LDA       #1
          JSR       [WRITE]
          PULS      U,DP
          JSR       ERROR
          RTS
 ******* Routine to Close File ********
SCLOSE    PSHS      U,DP
          LDU       #DCB
          JSR       [CLOSE]
          PULS      U,DP
          JSR       ERROR
          RTS
 *******Error Handling Routine *******
ERROR     TSTA
          BEQ       RETURN
          STA       $450
WAIT2     JSR       [POLCAT]
          BEQ       WAIT2
RETURN    RTS
 *** Memory for Buffers and Stacks ****
PBUF      RMB       256
LBUF      RMB       25
 *******Memory for Variables ********
DRVNUM    FCB       00
 ***********Memory for DCB ***********
DCB       EQU       *
          FCC       'WORKFILE'
          FCC       'TXT'
          RMB       38
 *********Memory for Message *********
MSG       FCC       'ENTER YOUR NAME:'
 ************************************
DONE      EQU       *
          END
```

Figure 10. Sample Program to Read to a File

Note: When running this program, a status code (generated by the Error subroutine) may appear on your screen. Press any key to continue program execution.

```
          ORG       $1200
 **Equates for DOS and ROM routines **
OPEN      EQU       $600
CLOSE     EQU       $602
READ      EQU       $604
POLCAT    EQU       $A000
CHROUT    EQU       $A002
 ****** Equates for DCB offsets *******
DEVNUM    EQU       $6F
SCREEN    EQU       0
DCBTRM    EQU       19
DCBDRV    EQU       33
DCBBUF    EQU       36
DCBLRB    EQU       39
 *****DOS Programming Convention *****
BEGIN     JMP       MAIN
          FDB       DONE-BEGIN
 ************Main Program ************
MAIN      JSR       CLEAR
          JSR       INTDCB
          JSR       SOPEN
          JSR       SREAD
          JSR       SCLOSE
          JSR       SPRINT
          CLR       $71
          JMP       [$FEEE]
 ******Routine to Clear the DCB ******
 and the Physical and Logical Buffers
CLEAR     LDX       #PBUF
CLEAR1    CLR       ,X+
          CMPX      #PBUF+255
          BNE       CLEAR1
          LDX       #LBUF
CLEAR2    CLR       ,X+
          CMPX      #LBUF+24
          BNE       CLEAR2
          LDX       #DCB+11
CLEAR3    CLR       ,X+
          CMPX      #DCB+48
          BNE       CLEAR3
          RTS
```

```
 ********* Routine to Insert **********
 ********* Values in the DCB **********
INTDCB    LDU       #DCB
          LDA       DRVNUM
          STA       DCBDRV,U
          LDA       #$ØD
          STA       DCBTRM,U
          LDX       #PBUF
          STX       DCBBUF,U
          LDX       #LBUF
          STX       DCBLRB,U
          RTS
 *******Routine to Open a File *******
SOPEN     PSHS      U,DP
          LDU       #DCB
          LDA       #$2F
          JSR       [OPEN]
          PULS      U,DP
          JSR       ERROR
          RTS
 *******Routine to Read a File *******
SREAD     PSHS      U,DP
          LDU       #DCB
          LDA       #3
          JSR       [READ]
          PULS      U,DP
          JSR       ERROR
          RTS
 ******* Routine to Print Data ********
SPRINT    LDB       #SCREEN
          STB       DEVNUM
          LDX       #LBUF
PRINT     LDA       ,X+
          JSR       [CHROUT]
          CMPX      #LBUF+24
          BNE       PRINT
WAIT1     JSR       [POLCAT]
          BEQ       WAIT1
          RTS
 ******* Routine to Close File ********
SCLOSE    PSHS      U,DP
          LDU       #DCB
          JSR       [CLOSE]
          PULS      U,DP
          JSR       ERROR
          RTS
 *******Error Handling Routine *******
ERROR     TSTA
          BEQ       RETURN
          STA       $450
WAIT2     JSR       [POLCAT]
          BEQ       WAIT2
RETURN    RTS
 *** Memory for Buffers and Stacks ****
PBUF      RMB       256
LBUF      RMB       25
 ********Memory for Variables ********
DRVNUM    FCB       ØØ
 ***********Memory for DCB ***********
DCB       EQU       *
          FCC       'WORKFILE'
          FCC       'TXT'
          RMB       38
 ************************************
DONE      EQU       *
          END
```